

Final Project: Advanced SQL Techniques

Estimated time needed: **60** minutes

Objectives

After completing this lab, you will be able to:

1. Use joins to query data from multiple tables
2. Create and query views
3. Write and run stored procedures
4. Use transactions

Scenario

In this project, you will work with three datasets that are available on the City of Chicago's Data Portal:

- Socioeconomic indicators in Chicago
- Chicago public schools
- Chicago crime data

Software Used in this Lab

In this lab, you will use [MySQL](#). MySQL is a Relational Database Management System (RDBMS) designed to efficiently store, manipulate, and retrieve data.



To complete this lab you will utilize MySQL relational database service available as part of IBM Skills Network Labs (SN Labs) Cloud IDE. SN Labs is a virtual lab environment used in this course.

Database Used in this Lab

Mysql_learners database has been used in this lab.

Here you will be creating and inserting data into the below mentioned 3 tables

- 1.chicago_public_schools
- 2.chicago_socioeconomic_data
- 3.chicago_crime

Here you will be using 3 dump files for this purpose.

[chicago_public_schools](#)

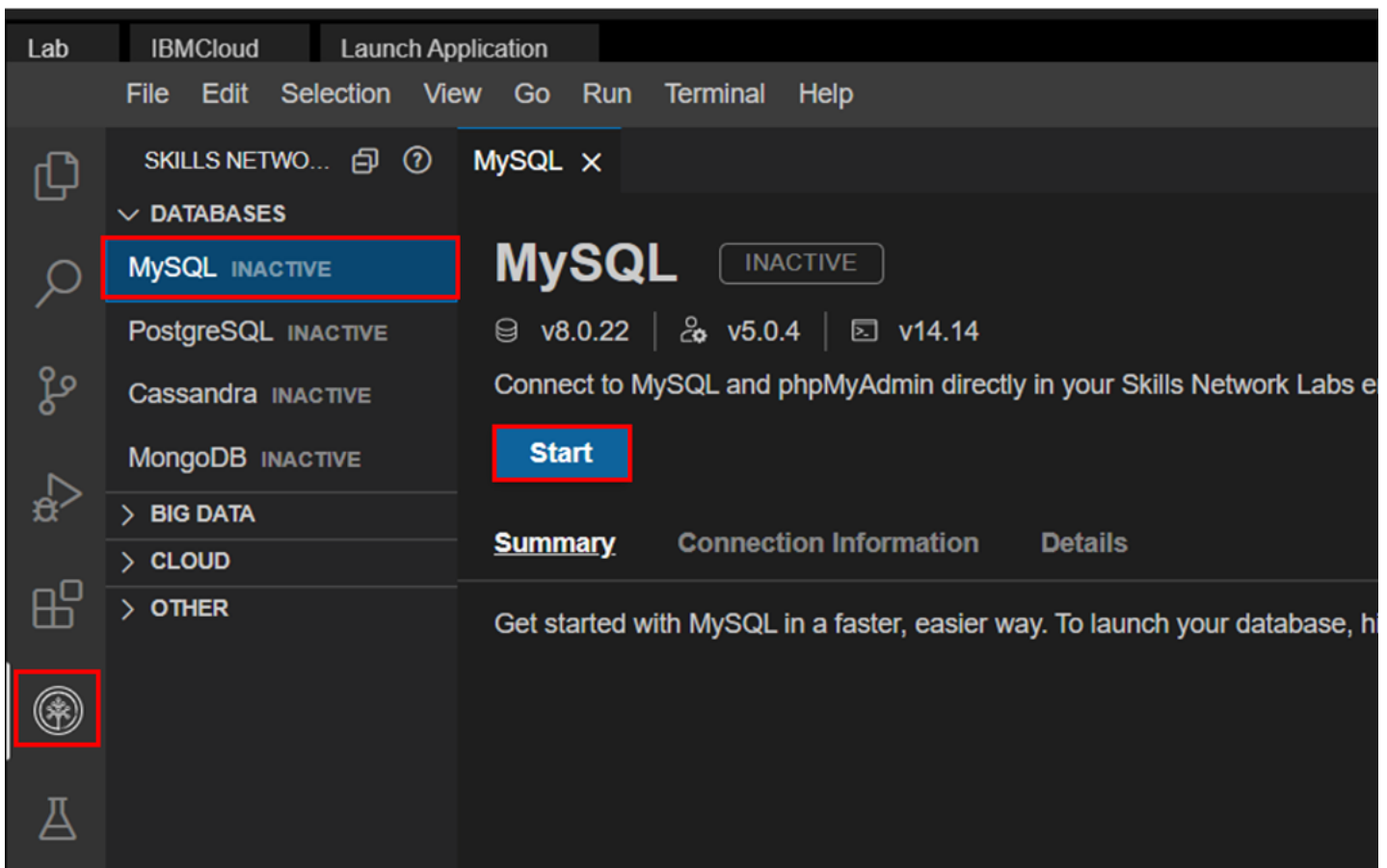
[chicago_crime](#)

[chicago_socioeconomic_data](#)

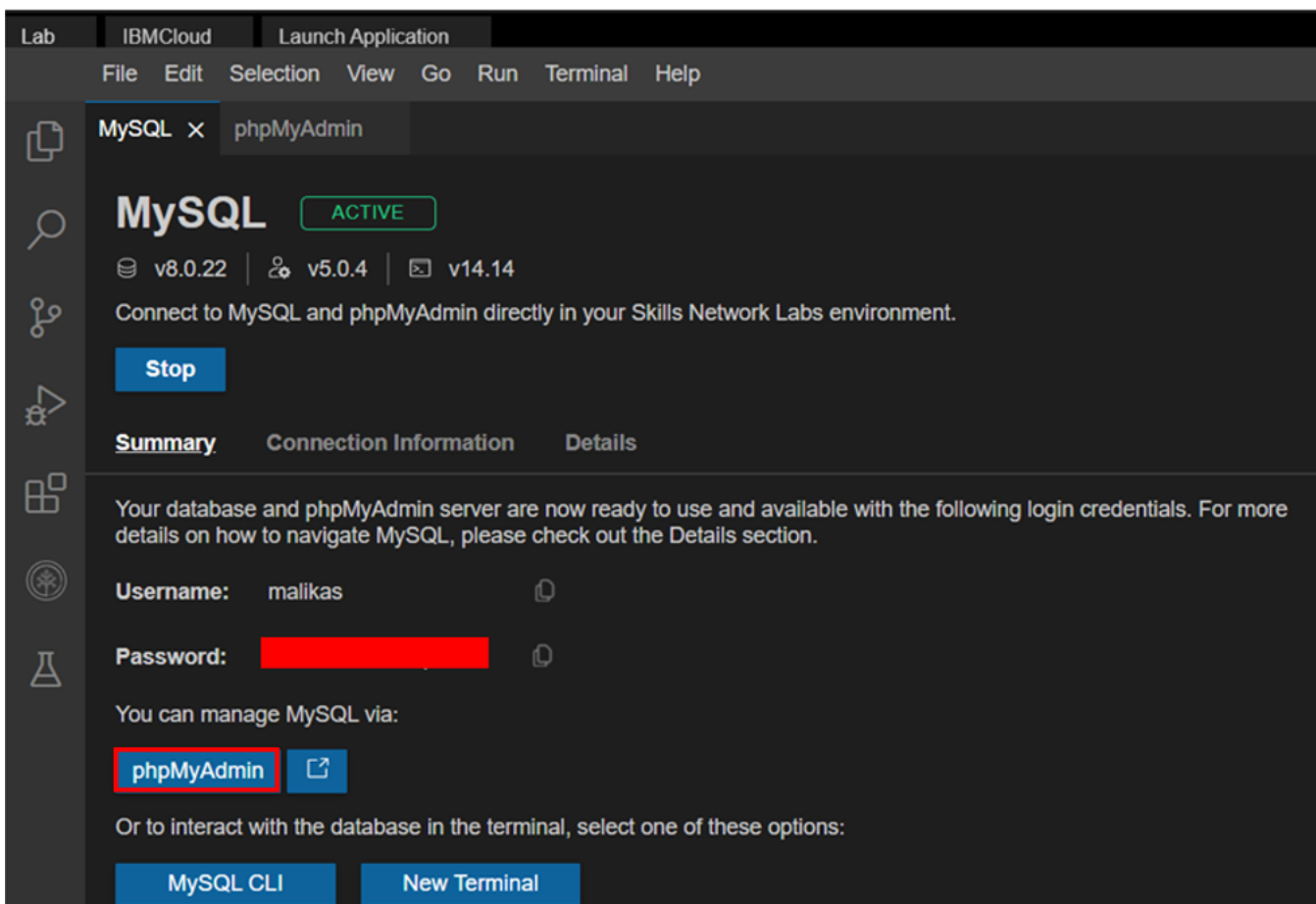
Task A: Create a database

1. Click on **Skills Network Toolbox**. In **Database** section, click **MySQL**.

To start the MySQL click **Start**.



2. Once **MySQL** has started, click on **phpMyAdmin** button to open **phpMyAdmin** in the same window.



3. You will see the phpMyAdmin GUI tool.



Recent

Favorites

- New
- information_schema
- mysql
- performance_schema
- sakila
- sys

Server: mysql:3306



Databases



SQL



Status

General settings



Server connection collation: ?

utf8r



More settings

Appearance settings



Language ?

English



Theme:

pmahomme ▼

4. In the tree-view, click **New** to create a new empty database. Then enter **Mysql_Learners** as the name of the database and click **Create**.

The encoding will be left as **utf8mb4_0900_ai_ci**. UTF-8 is the most commonly used character encoding for content or data.

Proceed to Task B.

Databases

SQL

Status

User accounts

Export

Import

Settings

Binary log

Repl...

Databases

Create database

Mysql_learners

utf8mb4_0900_ai_ci

Create

	Database	Collation	Master replication	Action
<input type="checkbox"/>	information_schema	utf8_general_ci	✔ Replicated	Check privileges
<input type="checkbox"/>	mysql	utf8mb4_0900_ai_ci	✔ Replicated	Check privileges
<input type="checkbox"/>	performance_schema	utf8mb4_0900_ai_ci	✔ Replicated	Check privileges
<input type="checkbox"/>	sys	utf8mb4_0900_ai_ci	✔ Replicated	Check privileges

Total: 4

⬆

☐ Check all

With selected:

Drop

Note: Enabling the database statistics here might cause heavy traffic between the web server and the MySQL server.

- Enable statistics

Load the dump files one by one into the database **Mysql_learners** by clicking the **Import** tab and choose the file. Click on **Go** button.

←

Server: mysql:3306 » Database: Mysql_learners

Structure

SQL

Search

Query

Export

Import

Operations

Privileges

Routine

Importing into the database "Mysql_learners"

File to import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**
Browse your computer:

Choose File

chicago_pu..._schools.sql (Max: 2,048KiB)
You may also drag and drop a file on any page.
Character set of the file:

utf-8

Partial import:

☒

 Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. *(This might be a good way to import large fi*
Skip this number of queries (for SQL) starting from the first one:

0

Other options:

☒

 Enable foreign key checks

Format:

phpMyAdmin

Recent Favorites

- New
- information_schema
- mysql
- Mysql_learners
 - New
 - chicago_public_schools
- performance_schema
- sys

Server: mysql:3306 » Database: Mysql_learners

Structure SQL Search Query Export Import Opera

✓ Import has been successfully finished, 22 queries executed. (chicago_public_schools.sql)

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0008 seconds.)

```
-- phpMyAdmin SQL Dump -- version 5.0.4 -- https://www.phpmyadmin.net/ -- -- Host:
Version: 7.4.15 SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO"
```

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```
START TRANSACTION
```

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

```
SET time_zone = "+00:00"
```

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0003 seconds.)

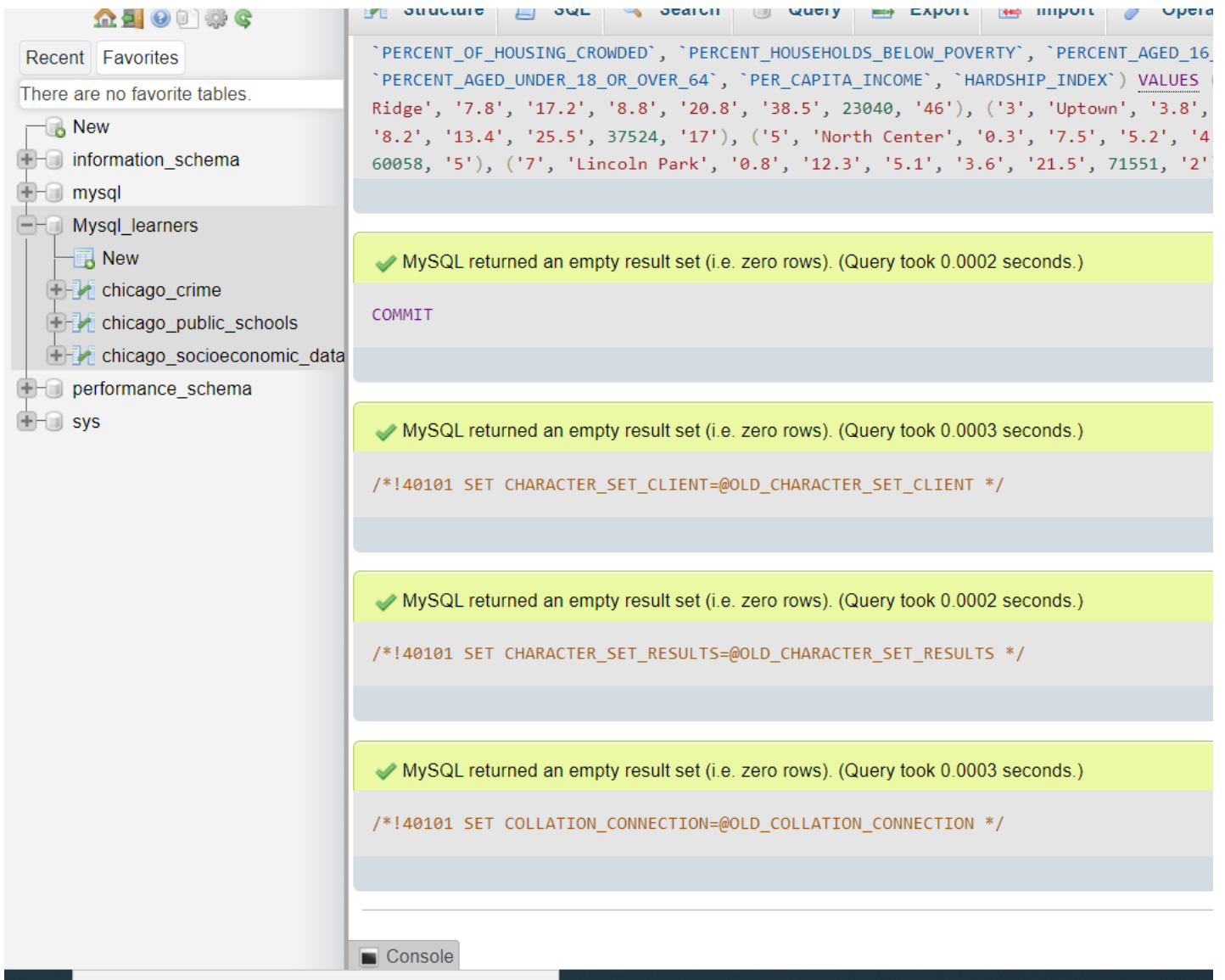
```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */
```

✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0005 seconds.)

```
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */
```

The tables are created and the data is loaded successfully. Repeat the same operation with the other 2 dump files to create and load the tables.

You will see a screen as below



Exercise 1: Using Joins

You have been asked to produce some reports about the communities and crimes in the Chicago area. You will need to use SQL join queries to access the data stored across multiple tables.

Question 1

- Write and execute a SQL query to list the school names, community names and average attendance for communities with a hardship index of 98.

▼ Hint 1

Use tables CHICAGO_PUBLIC_SCHOOLS and CHICAGO_SOCIO_ECONOMIC_DATA

▼ Hint 2

Use a left join

Take a screenshot showing the SQL query and its results.

Question 2

- Write and execute a SQL query to list all crimes that took place at a school. Include case number, crime type and community name.

▼ Hint 1

Use tables CHICAGO_CRIME_DATA and CHICAGO_SOCIO_ECONOMIC_DATA

▼ Hint 2

Use a left join

▼ Hint 3

The column location_description will help you find the crime location

Take a screenshot showing the SQL query and its results.

Exercise 2: Creating a View

For privacy reasons, you have been asked to create a view that enables users to select just the school name and the icon fields from the CHICAGO_PUBLIC_SCHOOLS table. By providing a view, you can ensure that users cannot see the actual scores given to a school, just the icon associated with their score. You should define new names for the view columns to obscure the use of scores and icons in the original table.

Question 1

- Write and execute a SQL statement to create a view showing the columns listed in the following table, with new column names as shown in the second column.

Column name in CHICAGO_PUBLIC_SCHOOLS	Column name in view
NAME_OF_SCHOOL	School_Name
Safety_Icon	Safety_Rating
Family_Involvement_Icon	Family_Rating
Environment_Icon	Environment_Rating
Instruction_Icon	Instruction_Rating
Leaders_Icon	Leaders_Rating
Teachers_Icon	Teachers_Rating

- Write and execute a SQL statement that returns all of the columns from the view.
- Write and execute a SQL statement that returns just the school name and leaders rating from the view.

Take a screenshot showing the last SQL query and its results.

Exercise 3: Creating a Stored Procedure

The icon fields are calculated based on the value in the corresponding score field. You need to make sure that when a score field is updated, the icon field is updated too. To do this, you will write a stored procedure that receives the school id and a leaders score as input parameters, calculates the icon setting and updates the fields appropriately.

Question 1

- Write the structure of a query to create or replace a stored procedure called UPDATE_LEADERS_SCORE that takes a in_School_ID parameter as an integer and a in_Leader_Score parameter as an integer.

Take a screenshot showing the SQL query.

Question 2

- Inside your stored procedure, write a SQL statement to update the Leaders_Score field in the CHICAGO_PUBLIC_SCHOOLS table for the school identified by in_School_ID to the value in the in_Leader_Score parameter.

Take a screenshot showing the SQL query.

Question 3

- Inside your stored procedure, write a SQL IF statement to update the Leaders_Icon field in the CHICAGO_PUBLIC_SCHOOLS table for the school identified by in_School_ID using the following information.

Score lower limit	Score upper limit	Icon
80	99	Very strong
60	79	Strong
40	59	Average
20	39	Weak
0	19	Very weak

▼ Hint 1

Remember that once a clause of the IF statement executes, no further checking occurs and processing moves to the code below the IF statement.

▼ Hint 2

Your IF statement should have a structure similar to:

```
IF in_Leader_Score > 0 AND in_Leader_Score < 20 THEN
```

```
-- update icon for 0-19
```

```
ELSEIF in_Leader_Score < 40 THEN
```

```
-- update icon for 20-39
```

```
ELSEIF in_Leader_Score < 60 THEN
```

```
-- update icon for 40-59
```

```
ELSEIF in_Leader_Score < 80 THEN
```

```
-- update icon for 60-79
```

```
ELSEIF in_Leader_Score < 100 THEN
```

```
-- update icon for 80-99
```

```
END IF;
```

Take a screenshot showing the SQL query.

Question 4

- Run your code to create the stored procedure.

Take a screenshot showing the SQL query and its results.

- Write a query to call the stored procedure, passing a valid school ID and a leader score of 50, to check that the procedure works as expected.

Exercise 4: Using Transactions

You realise that if someone calls your code with a score outside of the allowed range (0-99), then the score will be updated with the invalid data and the icon will remain at its previous value. There are various ways to avoid this problem, one of which is using a transaction.

Question 1

- Update your stored procedure definition. Add a generic ELSE clause to the IF statement that rolls back the current work if the score did not fit any of the preceding categories.

▼ Hint 1

You can add an ELSE clause to the IF statement which will only run if none of the previous conditions have been met.

Take a screenshot showing the SQL query.

Question 2

- Update your stored procedure definition again. Add a statement to commit the current unit of work at the end of the procedure.

▼ Hint 1

Remember that as soon as any code inside the IF/ELSE IF/ELSE statements completes, processing resumes after the END IF, so you can add your commit code there.

Take a screenshot showing the SQL query.

- Run your code to replace the stored procedure.
- Write and run one query to check that the updated stored procedure works as expected when you use a valid score of 38.
- Write and run another query to check that the updated stored procedure works as expected when you use an invalid score of 101.

Summary

You can now write advanced SQL statements to query data from multiple tables, to obscure sensitive data from users, and to control how information is updated in your tables.

Author(s)

[Lakshmi Holla](#)

[Malika Singla](#)

Changelog

Date	Version	Changed by	Change Description
2023-05-10	0.4	Eric Hao & Vladislav Boyko	Updated Page Frames
2023-05-04	0.3	Rahul Jaideep	Updated Markdown file
2022-07-04	0.2	Malika Singla	Updated screenshots
2021-11-01	0.1	Lakshmi Holla, Malika Singla	Initial Version

© IBM Corporation 2023. All rights reserved.